NASA-CR-200662

[illegible]

A Collaborative and Concurrent Environment for IV & V

Phase 3

SemiAnnual Report

submitted by

West Virginia University

Phase 3 Report

Task 1

1. Task 1 - IV&V Process Improvement

1.1 Summary

The objective of this task is to investigate methods by which the IV&V process can be improved. During this phase the goal of this task was to define a collaborative IV&V process to address the constraints in performing IV&V for NASA's projects.

The collaborative IV&V process is defined utilizing the Capability Maturity Model (CMM) framework, concepts of peer reviews and software estimation principles. The process model addresses constraints in performing IV&V discussed in Section 1 of the previous report.

1.2 Work Completed

Defined a collaborative IV&V process model.
Identified the roles in the collaborative IV&V process and their responsibilities.
Identified key events in the collaborative IV&V process.

1.3 Review of Constraints on IV&V process

The constraints in performing IV&V are due to the Verification and Validation cycle, NASA's project features and dependency of IV&V process on software development process. The verification and validation cycle imposes constraints like the need to track the responses to problem reports and maintaining the flow of information across different phases of the life cycle. NASA's projects are of long duration, involve multiple contractors and mission critical software. Long project duration results in the evolution of requirements and an unstable work group. Unstable work group composition results in the loss of information and hence acts as a bottleneck for IV&V activities. When multiple contractors are used for development, the IV&V process should interact with different processes at different levels of process maturity. Mission critical software demands a very high quality work necesitating continuous process improvement. Dependency of IV&V process on software process results in the need to tailor IV&V process for the software process. With multiple contractors there is a need to interact with different process, methodologies, and maturity levels.

1.4 Collaborative IV&V process Model

The first step in defining a collaborative process is to ensure compatability between software development process and IV&V process. The Capability Maturity Model (CMM) is emerging as a standard process for software development and is actively sponsered by the government agencies like the DOD, USAF etc. The DOD requires its contractor be at level 3 in the maturity scale in order to compete for certain contracts. Hence a level 3 process in the maturity scale is used as a reference to define the collaborative IV&V process. The collaborative IV&V process model utilizes the concepts of peer reviews as a means of collaboration. Bottom Up method for estimation and tracking are utilized for the purposes of cost and schedule estimation. The following sections describe the collaborative IV&V process model in detail.

1.4.1 Overview of Collaborative IV&V Process Model

The collaborative IV&V process is based on a level 3 process in the maturity scale. Significant features of a level 3 process are:

- Organization process focus.
- Organization process definition.
- Peer reviews.
- Intergroup coordination.
- Software product engineering.
- Integrated software management.
- Training program.

In the collaborative IV&V process model, the organization process focus is retained as defined in the maturity model. The concept of process definition is modified. In the maturity model, process definition is derived from process focus by the individual project teams. In the collaborative IV&V process model, process definition is derived from the process focus by the individual tasks. Since individual tasks interface with different software contractors, defining process definition at the task level would ensure the differences in the U-level process of software contractor and hence the differences in interfacing with the software contractor could be absorbed in the process definition. As a result of this variation, the definition and the extent of organization process focus and process definition is modified. A comparison of the maturity model and that adapted for IV&V process model is shown in Figure 1.1. CMM 1.1 [2] defines process focus as a typical process for developing and maintaining software. Humphrey [3] provides a list of categories for process focus:

- Management and Planning standards.
- Development process standards and methods.
- Tool and process standards.

In the collaborative IV&V process model, organization process focus is comprehensive but not very detailed. This allows the required flexibility for tasks to suitably define the process definition. Process definition are derived at task level so that the software process features, maturity level of the process, etc. is considered in deriving the process definition. Hence the process focus will be a general set of guidelines for all the tasks.

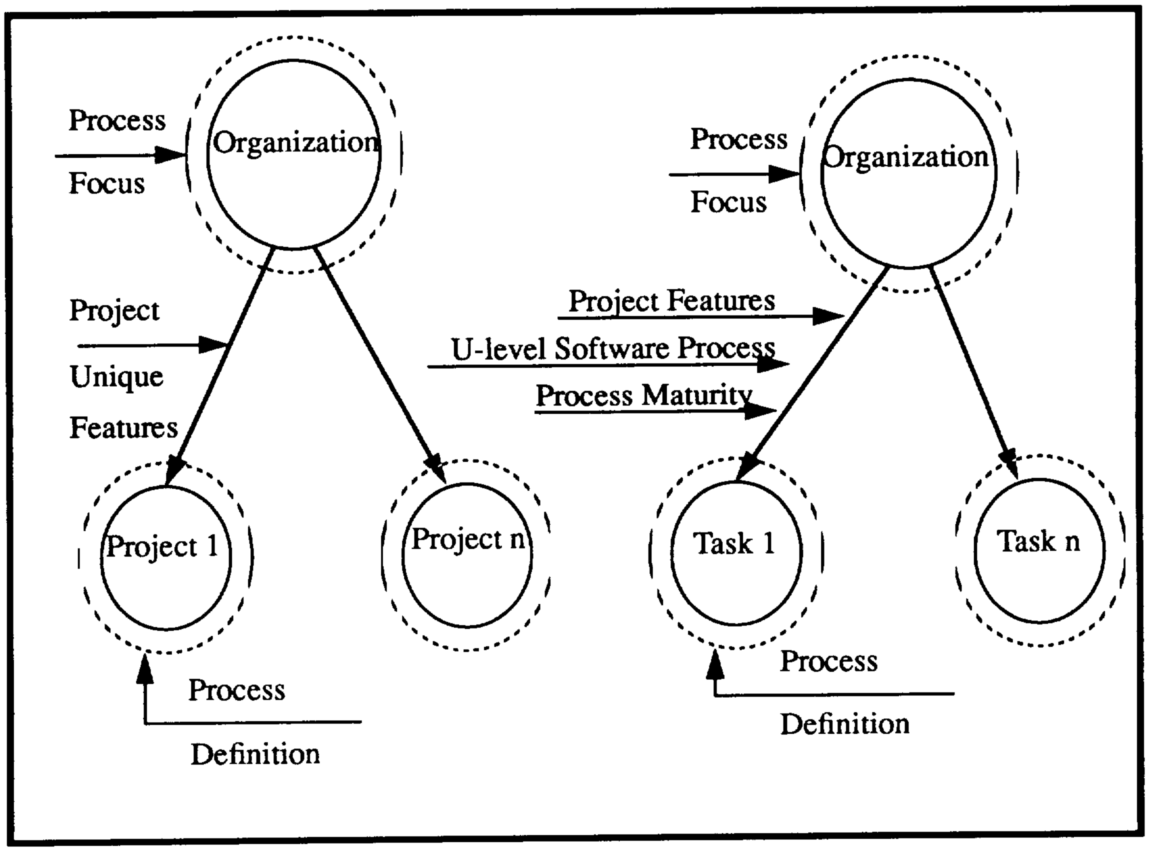


Figure 1.1 Comparison of Maturity model and its IV&V Adaptation

Aspects that will be covered by process focus are:

- Guidelines on tool sharing with the software contractor.
- Guidelines on information sharing with the software contractor.
- Guidelines on document verification.
- Guidelines on mailing reports.

At the task level, process definition will be derived within the framework provided by the process focus. While defining the process definition, project features, the software process feature of that software contractor whose products are verified and validated by this task, the process maturity of that contractor are all considered. Hence at the task level the IV&V process and software process are compatible.

Software inspections have proved to be effective in detecting errors not only in the coding phase but also in other phases of software development life cycle [5]. Software inspections are also a part of level 3 process in the maturity scale. Hence software inspections are utilized as a validation method in the IV&V process. Since this is part of a level 3 process in maturity scale, the software contractor is expected to utilize software inspections for detecting errors. In a collaborative environment the IV&V team would participate in software inspections organized by the software contractor, and the feedback from the IV&V team would be direct and immediate.

Since the primary objective of using an IV&V contractor is to provide an unbiased perspective of the product, the participation of IV&V in software inspections organized by software contractor should not involve face to face interaction with the software contractor. A software inspection tool could be used. The use of a tool would result in IV&V reviewing the products without any direct interactions with the software contractor.

In the collaborative IV&V process model, the bottom up method is used for both cost and schedule estimation. The Personal Software Process (PSP) from SEI also uses a bottom up approach for software process improvement [4]. The emphasis of first phase PSP or PSP0 is to let the engineers estimate metrics like development time, cost, etc. for their segment of work. At the next level of PSP, engineers use their historical database of estimates to improve the accuracy. In the collaborative IV&V process model, members of the IV&V organization make an initial estimate of the development cost and time. After a certain time, the initial estimates are baselined i.e. the initial estimates cannot be updated after this time. Periodically, members of this organization would report the amount of work completed by them. The amount of work completed would be used to calculate the equivalent man months of work completed. The man months of work completed is compared with the initial estimates to determine the current status. Thus the bottom up approach for estimation is extended to be an effective project tracking mechanism.

1.4.2 Organizational Hierarchy

The organizational hierarchy and the roles in this hierarchy are defined in the following paragraph. Figure 1.2 depicts the organizational hierarchy.

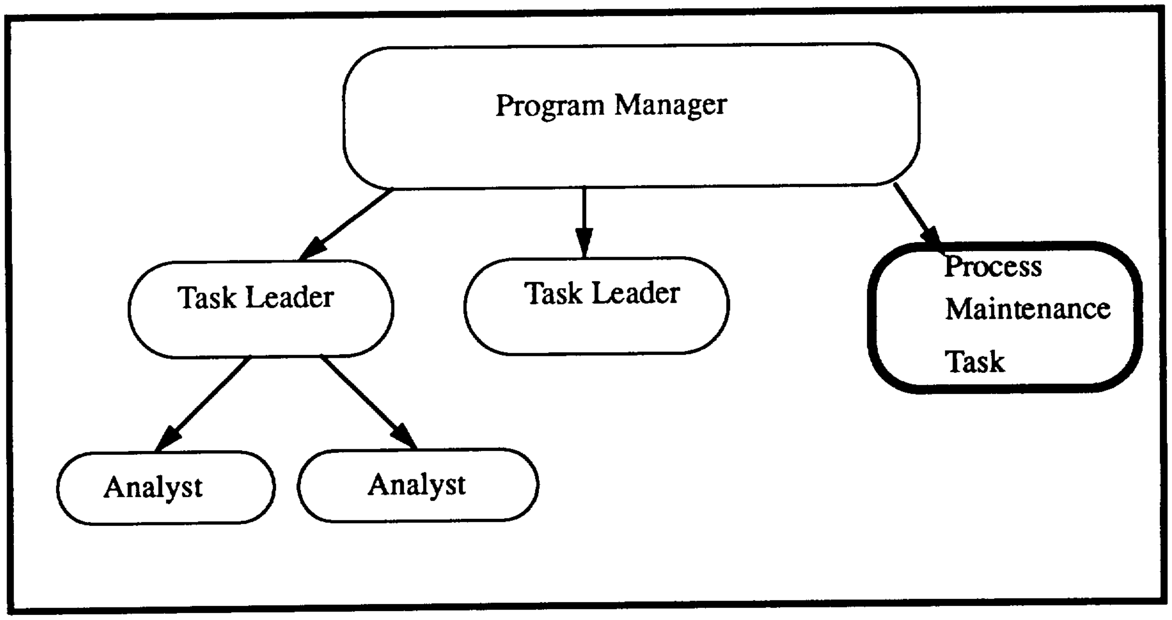


Figure 1.2: Organizational Hierarchy

The program manager is responsible for the project organization, project performance, task identification, allocation of resources for tasks and the organization process focus. Task leaders are responsible for task performance, task organizing, providing technical direction for the task and also the process definition at task level. The analyst perform the task allocated to them by the task lead, and populate the IV&V analyst notebook. The process maintenance task is essentially the equivalent of the Software Engineering Process Group (SEPG) defined in the maturity model. This task maintains the process focus, process definition, process metrics, the task structure and the IV&V personnel database. The key events in this process are requirements change, additional task requested by the customer and change in work group composition. The responsibilities of the roles identified and the key events are elaborated in Section 1.4.

1.4.3 Relationship between the Collaborative IV&V process model and the Organizational hierarchy

The relationship between the collaborative IV&V process model defined in Section 1.4.1 and the organizational hierarchy identified in Section 1.4.2 will be discussed in the following paragraphs. The relationship is highlighted by identifying the additional responsibilities of the roles with respect to the process model defined. This set of responsibilities is a subset of the responsibilities listed in Section 1.4.4.

Additional Responsibilities of the Program Manager

The program manager is responsible for the process focus. The process focus is uniformly utilized by the project tasks for deriving the process definition. It is the responsibility of the program manager to review the changes requested by the members of this organization and incorporate the changes into process focus.

Additional Responsibilities of the Task Leader

The task leader's primary responsibility is to derive the process definition within the framework provided by the process focus. The task leader reviews the changes requested by task members and incorporates the changes into process definition. Task leaders provide development cost and schedule estimates.

Additional Responsibilities of the Analyst

The analyst participate in the peer reviews organized by both the software contractor and the IV&V contractor. Analyst populate the IV&V analyst notebook. They also provide initial estimates for development cost and time, and periodic estimates for the amount of work completed.

Additional Responsibilities of Process Maintenance Task

The primary function of the process maintenance task is to maintain the process focus, process definition, identify and maintain process metrics. This task also interacts with the task leaders to ensure that process definition is defined within the framework provided by the process focus.

1.4.4 Responsibilities and Key Events in the Collaborative IV&V Process

The responsibilities of the different roles in an IV&V organization is discussed in this section. The key events in this process are also discussed. To begin with, the responsibilities are discussed as follows:

- Program Manager
- Task Leader
- Analyst
- Process Maintenance Group

1.4.4.1 Program Manager

The program manager is responsible for the entire project. The program manager is not only responsible for the technical performance but also for the fiscal performance. The task leaders and the process maintenance group report to the program manager. The responsibilities of the program manager are:

[a] Define Organization Process Focus: As defined in CMM 1.1 [2], process focus is a document that defines a typical process for developing and maintaining software across the organization. The definition conveys the idea that process focus is a set of policies and guidelines that could be used by different project teams for deriving a set of actions and methods for a specific project. Since the process focus is the basis for process definition, similarity in process definition across different project teams could be expected. Hence the process focus should be comprehensive, not only in providing guidelines covering different aspects, but also identify a set of process metrics to measure quality, status, etc.

[b] Define Project Tasks: The program manager identifies tasks and the respective task goals. The program manager along with the process maintenance group identify and assign task leaders for the tasks. This activity is of paramount importance as task leaders are responsible for defining process definition, populating the task and also providing technical directions. Program manager should also allocate the required resources for the tasks.

[c] Oversee Project Tracking: Project tracking is the process of reviewing the current status and results against the initial estimates. This allows the program manager to take corrective actions when the performance of a task deviates from the estimated performance. Changes in requirements, task composition, etc. should be considered before determining the corrective action.

[d] Act as a contact point for the entire project: The program manager interfaces with the customer and software contractor. Program manager is responsible for defining the goals of the IV&V contract. The program manager is also responsible for reporting the fiscal and technical performance to the customer.

1.4.4.2 Task Leader

Task leaders are responsible for their tasks. Task leaders report to the program manager. The responsibilities of a task leader are:

[a] Define Process Definition: The task leader should define the process definition for the task. Process definition should be defined within the framework provided by the organization process focus. Task leader should consider the task goals, process focus, the software process, and maturity level of the software contractor in defining the process definition. A process definition should include software engineering methods like metrics collection, management methods like

estimation, technical methods or approach to be used, the inputs needed for the task like requirements document, outputs to be generated like problem reports, and also a schedule for the task.

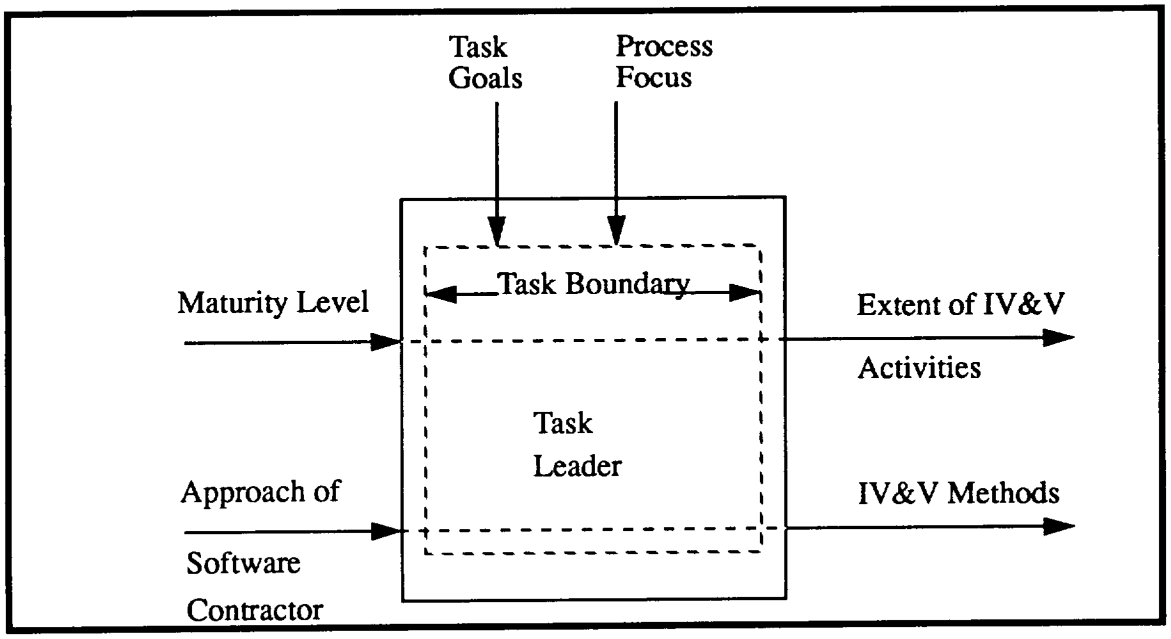


Figure 1.3 Role of Task Leader

As depicted in Figure 1.3, the task goals and the process focus serve as task boundary for each task. The maturity level of the organization determines the extent of certain activities performed by the task [1]. The software contractors methods and approach have an impact on the IV&V methods and approach. This approach for process definition ensures that the differences in the maturity level of software contractors are taken into consideration at the task level. Also this approach for task definition ensures that the IV&V methods are suitably tailored to the software process and methods of the software contractor. Hence defining process definition at task level ensures that issues relating to interfacing with more than one software contractor is effectively addressed.

[b] Act as Contact point for the task: The task leader is responsible for defining task goals, methods, approaches, and also for tracking task status. The task leader has a complete understanding of the nature of the task and the current status.

In the organizational hierarchy currently used by IV&V organizations, a separate group is entrusted with the responsibility of maintaining information flow. A possible reason for this is that an IV&V organization should follow certain guidelines for sharing information as stipulated by the customer. In the collaborative IV&V model this constraint is addressed by making the guidelines for information sharing a part of the process focus. Hence these guidelines would also form a part of the process definition of individual tasks. The task leaders then can act as contact points for any information regarding the task. This structure is depicted in Figure 1.4b.

[c] Estimate the resources needed: After defining the process definition for the task, the task leader should estimate the resource requirements for the task. Task leaders should estimate the man months of effort required and also the tool requirements.

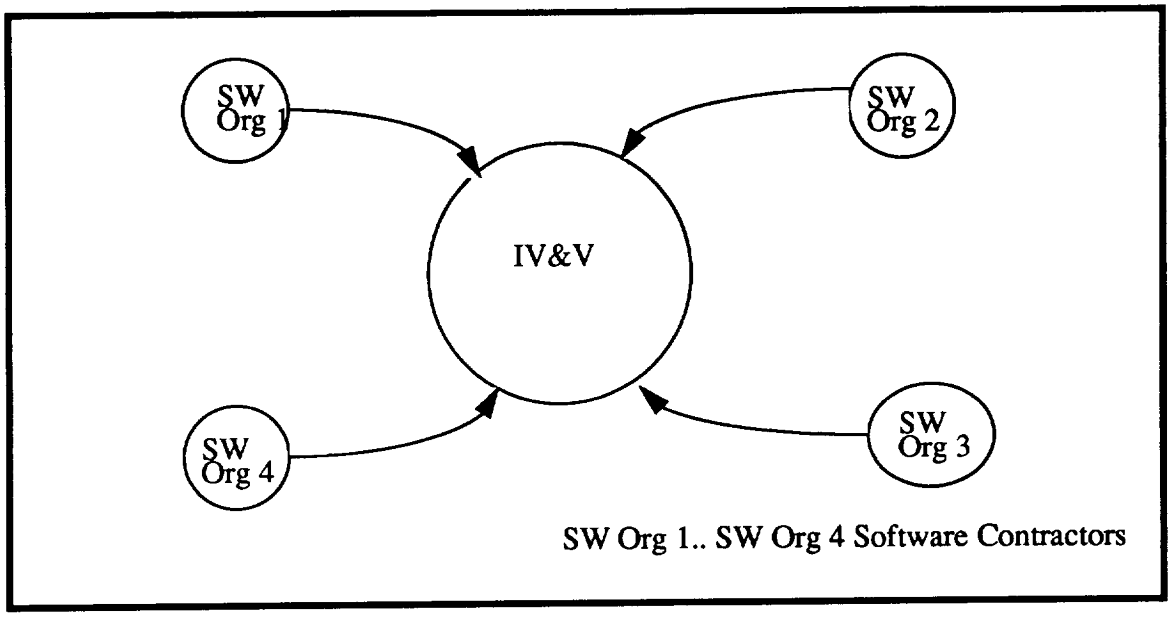


Figure 1.4a Effect of Multiple Interfaces on IV&V

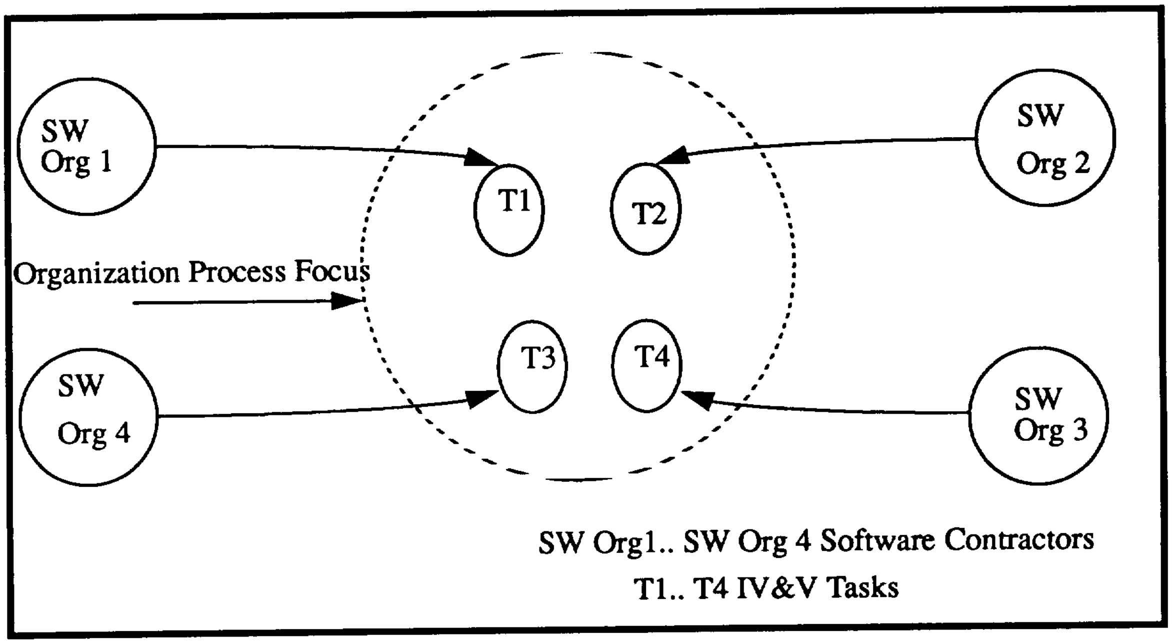

SW Org 1
SW Org 2
T1
T2
Organization Process Focus
T3
T4
SW Org 4
SW Org 3
SW Org1.. SW Org 4 Software Contractors
T1.. T4 IV&V Tasks

Figure 1.4b Proposed Approach

[d] Populate Task: Once the process definition for the task is defined, the task leader should identify analyst with suitable skills to populate the task. Identifying analyst with the right skills is crucial to the success of the task.

[e] Provide Technical Direction: The task leader should define the approach for performing the different V&V tasks. The task leader is responsible for defining methods, the inputs needed for

the task and also the output that should be generated by this task.

1.4.4.3 Analyst

The activities of an analyst are listed below:

[a] Estimate the development time and cost for performing the allocated work: The analyst is expected to provide an estimate of the development time and cost for the allocated work. This estimate is used as a reference to track the status of the task during the development life cycle.

[b] Participate in peer reviews: Peer reviews are used to detect defects in the products developed during the life cycle. The analyst can play the role of a producer by requesting for peer review, can volunteer to coordinate a review requested by a peer or can volunteer to review the products developed by a peer.

[c] Report the amount of work completed: Periodically the process maintenance group would request for the amount of work completed by the analyst. The analyst provides the amount of work completed as a percentage of the total work. This input is used to generate metrics on the status of each task.

[d] Populate IV&V Analyst note book: The IV&V analyst note book is used to store comments on the activities performed by the analyst. This information will be of use during the subsequent phases.

[e] Perform allocated segment of the task: The IV&V analyst performs the allocated segment of the task using the methods and approach defined by the task lead.

1.4.4.4 Process Maintenance Group

The key functions of the process maintenance group are listed below:

[a] Maintain Organization Process Metrics: The process maintenance task should work with the program manager to define the process focus. Once the process focus is stable after the initial iterations, the process maintenance group should baseline the process focus. Baselining process focus signifies that the process focus is stable and the task leaders can derive process definition based on the process focus. During the development life cycle the members of the organization might request for changes in the process focus based on their experience in using the process focus. The process maintenance group should review these suggestions and incorporate them into process focus at a suitable time.

[b] Assist program manager in identifying and defining tasks: The process maintenance group assists the program manager in identifying or dividing the project into tasks, and defining the goals for each task. This set of goals is used as a basis for defining the process definition by the task leaders.

[c] Maintain Process Definition: Task leaders define process definition for their tasks. The process maintenance group ensures that the process definition is within the framework provided by the process focus. The process maintenance group baselines the process definition once the definition

is stable. After baselining, the process definition cannot be updated even by the task leaders. Members of the task can suggest changes to the process definition based on their experience in using the definitions. The process maintenance group should update the process definition to incorporate these changes at a suitable time.

[d] Maintain Process Metrics: Process metrics are measures of software development process. Process metrics are used for efficient management of software development process [6]. The process maintenance group should identify suitable metrics that reflect the parameters of software development and periodically update these metrics. Periodic update of process metrics will provide a clear visibility of the status for the program manager and the task leaders.

<u>1.4.4.5 Key Events in the Collaborative IV&V Process Model</u>

The key events in the collaborative IV&V process are:

[a] Requirements Change: The changes in requirements was identified as a result of long project duration. It is of considerable importance that the changes in the requirements is tracked and addressed. In the following section a sequence of activities are proposed to address requirements change.

The sequence of activities are depicted in Figure 1.5 using the Entry, Task, Exit and

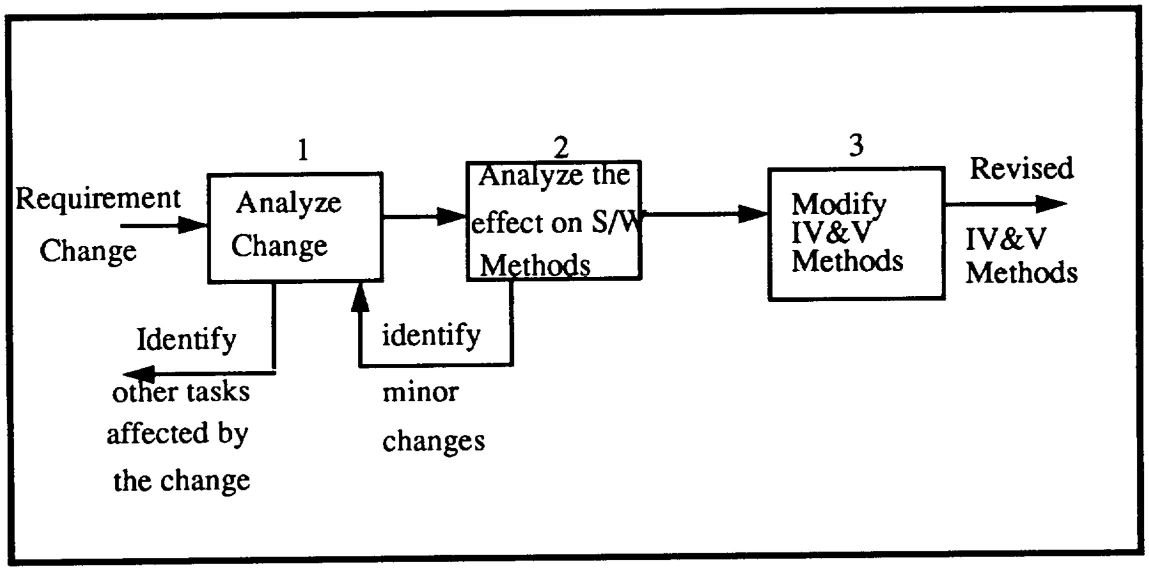


Figure 1.5: Requirement change sequence

Measure (ETXM) specification. When using this specification a sequence of actions that should be performed under the event is identified. Then for each of the activity in the sequence, an entry criteria, action, exit criteria and a measure are identified. When the entry criterion is met the action is initiated. Once the action is initiated the activity described is performed until the exit criterion is met. The measure identifies the performance during the sequence.

The following table lists the ETXM criteria for the sequence of activities due to requirement change.

Table 1.1 Requirement Change Sequence

Cell	1	2	3
Entry	Change in requirements	Changes in requirements analyzed	Changes in S/W contractors methods analyzed
Exit	Changes in requirements analyzed	Consequent changes in methods by S/W contractor analyzed	Revised IV&V methods
Feedback in	Minor changes in requirements		
Feedback out	Identification of other tasks	Minor changes in requirements	
Task	Analyze requirement change	Analyze changes in S/W methods	Revise IV&V methods
Measures	Number of tasks affected Extent of Impact	Extent of change in methods	Changes in resource requirement

Once a requirement change is notified, the effect of the change should be analyzed. As a result of this activity the IV&V tasks directly affected by this requirement change should be identified. The tasks affected by this change, analyze the change, its impact on software methods and its effect on IV&V approach and methods. As a result of this analysis other tasks affected by this change are also identified. Based on the analysis performed in the previous step, IV&V methods are modified to address the changes. A set of measures is identified in the table. These measures act as an indicator of the impact of the change and hence on the performance, and resources of the affected tasks.

[b] Additional Task Request: Additional work may be requested by the customer as and when needed. As a result of this request, new tasks should be defined, organized, and resources should be allocated and integrated with other tasks. The program manager along with the process maintenance group should define the goals for this task. A task leader should be assigned to the task. The task leader along with the process maintenance group should define the process definition for this task. Defining process definition at the task level helps in addressing this problem as well. In a typical process the process definition is performed at the project level, where a request for a new task results in rearrangement of resources without clear planning. Defining process definition at the task level addresses this constraint directly.

[c] Dynamic Work Group Composition: Changes in work group composition are inevitable. The reasons for this change are:

- Personnel leaving the organization.

- Addition of new personnel.
- Organizational rearrangement.

The primary concern in a dynamic work group environment is that the experience gained by the analyst is lost when the analyst moves out of the task. The use of IV&V analyst note book is advocated in this scenario. The insight gained by the analyst is stored in the analyst note book by way of comments. Advocating the use of software inspections, Boehm [5] states, "It (software inspections) also has significant side benefits in team building and in ensuring backup knowledge if a designer or programmer leaves the project". Thus the use of IV&V analyst note book and software inspections ensures that the knowledge gained by an analyst is stored in the IV&V analyst note book or disseminated among peers. Hence the dependence of the IV&V process on IV&V analyst is minimized. The use of a organization process focus address the issue of rearrangement, since process definition of all the tasks are based on the same process focus, personnel should be trained only on task specific methods. When adding a new analyst, the analyst should first be trained in the process focus and assigned to a particular task. The analyst should next be trained in task specific methods.

1.5 Future Work

During the next phase this task will analyze the need for an environment to support the collaborative IV&V process. A suitable architecture that would address the constraint like geographically distributed development, hardware environment encompassing different platforms like PC's, Workstations etc will be designed and implemented.

1.6 References

[1] Robert O. Lewis, Independent Verification and Validation, A Life Cycle Engineering Process for Quality Software; John Wiley & Sons, Inc., 1992.

[2] Mark C. Paulk, Bill Curtis, and Mary Beth Chrissis, "Capability Maturity Model Version 1.1", IEEE Software, July 1993.

[3] Watts S. Humphrey, Managing the Software Process, Addision-Wesley Publishing Company, 1989.

[4] Roger S. Pressman, "Making Process Improvement Personal", IEEE Software, pp 82 - 83, September 1995.

[5] Barry W. Boehm, "Industrial Software Metrics top 10 list", IEEE Software, pp 84 - 85, September 1987.

[6] Everald E. Mills, Software Metrics SEI Curriculum Module SEI-CM-12-1.1, Software Engineering Institute, Pittsburgh, 1988.

Phase 3 Report

Task 2.1

1. Task 2.1 - Infrastructure Framework for IV & V

Software Pro ject Management and Measurement on the World-Wide-Web (WWW)*

Abstract

We briefly describe a system for forms-based, workflow management that helps members of a software development team overcome geographical barriers to collaboration. Our system, called the Web Integrated Software Environment (WISE), is implemented as a World-Wide-Web service that allows for management and measurement of software development projects based on dynamic analysis of change activity in the workflow. WISE tracks issues in a software development process, provides informal communication between the users with different roles, supports to-do lists, and helps in software process improvement. WISE minimizes the time devoted to metrics collection and analysis by providing implicit delivery of messages between users based on the content of project documents. The use of a database in WISE is hidden from the users who view WISE as maintaining a personal "to-do list" of tasks related to the many projects on which they may play different roles.

Keywords: Workflow management, verification and validation, software engineering, issue tracking, software measurement, software metrics, software process

1.1 Introduction

Collection of metrics and adherence to a disciplined development process are difficult tasks in any software project. Yet, project developers and managers need to understand their processes in order to coordinate assigned tasks effectively. To understand their own work processes and assess the status of an ongoing project, they must be able to measure various aspects of the project tasks, people, and products. An assessment of a project's status is critical to the reassignmen of resources, adjustment of schedules, and measurement of product quality. Change activity is a powerful indicator of a project's status. Automated systems that can handle change requests, track issues, and process electronic forms provide an excellent platform for tracking the status of the project. We have developed a World-Wide-Web-based approach called the Web Integrated Software Environment (WISE) that supports measurement of change activity as an implicit part of the software process. WISE provides a forms-based, workflow management system that helps members of a software development team overcome geographical barriers to collaboration. WISE allows for the improvement of the software process in a realistic environment based on dynamic analysis of changes to information and communication in the workflow. WISE tracks issues in software development process, provides informal communication between the users with different roles, supports to-do lists, and helps in software process improvement. WISE minimizes the time devoted to metrics collection, analysis, and reporting tasks not related directly to project activities. Automated tools like WISE focus on understanding and management of a software process rather than the bureaucracy of an organization.

Since the summer of 1995, WISE has been used on several projects within the National Aeronautics and Space Administration (NASA) and the private sector to coordinate and monitor software verification and validation (V&V) activities. This paper briefly discusses issues related to the use of WISE in the automation of software project management and measurement. From our practical experiences with WISE, we believe that such automated tools can transform chaotic software development projects into more controlled and manageable processes.

1.2 Overview

WISE can be installed and made available at a specific URL (Universe Resource Locator) on the Internet or within a corporate network. Using an appropriate browser (WISE requires support for HTML 2.0 tables. The Beta and production releases of WISE also require support for Java. Many WWW browsers support tables including the most recent versions of NetScapetm and Mosaic. The use of arbitrary HTML allows WISE to be easily integrated with other WWW-based tools, documents, and resources on a software project. For example, Figure 1 shows issue connections to Review Item Discrepancy (RID) documents on another WWW server. We have also used this facility to connect WISE almost effortlessly to the NCSA prototype HyperNews tool.) each user connects into the WISE home page to view their personal "to-do lists" (TDLs) available on several projects. Each person in an organization may be assigned to one or more projects and therefore may have one or more TDLs under WISE. Figure 1 shows a portion of a person's "to-do list" for a sample project. Each item on a person's TDL is called an issue. Each issue contains a set of fields and associated values. These fields are a superset of those shown for each issue on the TDL. Individual issues may be viewed by selecting them as hyperlinks on a TDL. For example, Figure 2 shows an individual issue (i.e., issue number 2) from the TDL shown in Figure 1. WISE TDLs and issue forms can be customized for specific organizational needs. For example, a WISE application ca be configured to maintain a list of tasks, an order-entry database, problem reports, or phone contact lists. Legal types of fields in an issue can include: free-form text, finite selections (single and multiple), hypertext (i.e., arbitrary HTML2), numbers (integer, real), dates and times, radio buttons and check boxes.

An issue appears on an individual's TDL based on their role(s) on projects. Roles are created and configured by project administrators to control information access. A user may play more than one role on a project, but the visibility of fields in an issue and the types of changes allowed on an issue form are dependent on the user's role(s). Roles act as filters on all project issues. This approach provides users with a unique view of their tasks on a project. Changes to an issue can be submitted by a user directly through the WWW browser because issues are implemented using the CGI Forms Interface. Once submitted, the issue changes are recorded in a log and the issue may disappear from the user's TDL and appear on the TDLs of other users. The specific workflow of issues within an organization is dependent on the customized definitions of issues, fields, and roles on a project in the WISE site configuration file. WISE also provides for delivery and submission of forms via electronic mail to accommodate user's between Internet firewalls, but such users must read the mail via an HTML capable browser. Thanks to the ubiquitousness of WWW browsers, users can access WISE through the World-Wide-Web from a variety of hardware platforms and operating systems. WISE also provides on-demand access to project metrics. WISE keeps track of changes to issues and other project events. WISE collects metrics based on these events and presents graphical views of the project statistics. In one project, for example, issues can track problems discovered during development. Issues can cycle from an open status to a solved or closed status. Issues marked as solved are those problems that have been diagnosed but not fixed. Once a problem is fixed, the issue can be marked closed. Figure 3 depicts metrics for a user's project plotted over time. It shows that the number of open issues initially exceeds the number of closed issues and that very few issues are recorded as "solved" before being closed. Furthermore, this organization noted that the trend in open vs. closed issues helped to estimate the time of their first software release as the projected time when the number of open issues fell below the number of closed issues WISE is non-intrusive because it provides a "to-do" list of each issues to each developer in the team. Each issue in the "to-do" list can be acted upon by changes to the values of fields in an issue. The types of changes allowed are dependent on a user's assigned role(s). The composition of the forms and views defines the totality of the software process. Thus, the process is not fixed or globally defined by the manager, but it is highly dynamic and may change based on the different roles of development personnel throughout the lifecycle of a development effort.

1.3 Process Improvement

In order to achieve the goals of any software project, one must be able to assess the quality of a development process itself. We must be able to improve the software process continually and determine if it is progressing at an acceptable rate. To produce quality products and improve the capability of the organization to produce better products, the software process must improve as well. Software maturity frameworks are usually characterized by different levels in which an orga- nization can be categorized depending upon the results of the assessment of the organization's software process [7]. Many case studies have been conducted and have shown that there is a need to turn the corner from chaotic, unpredictable cost to a more manageable and controlled software process whereby schedule slippage and cost overruns are avoided. Automated support of measurement plays an important role in software process improvement. Project managers rely on a range of methods for measurement including status reporting and change management. Handling change requests, problem reports, activity log entries, and other issues in a software development effort becomes quite complicated even in small groups. We believe that automation plays a key role in increasing productivity, controlling quality, and introducing predictability into the software process. But the effective use of software technology is limited by ill-defined processes and poor process management. These problems must be addressed for us to be able to apply new tools and technology. Much research in this area has shown that incremental automation in software management and measurement is necessary to achieve successful software process improvement. The focus of more recent approaches to process improvement has been through analysis and synthesis of organizational experience. Information collected about a workflow in a repository of such experience can be analyzed for specific organizations to help them improve weak areas and capitalize on strengths. The Experience Factory mechanism [2] is an approach to improving software development processes. The Experience Factory packages the experiences of an organization and measures various software process and products. WISE can be used to collect valuable information related to issue resolution in a database implicitly. This information sheds light on the issue solving capacity of a development group and its products. For example, modules with the most severe issues and the average time for solving issues are two statistics that can be obtained by analyzing software process change data.

1.3.1 Measurement

Software metrics can be defined as the continuous application of measurement-based techniques to the software development process and its products. Metrics supply meaningful and timely management information used to improve a process and its products [12]. Software metrics are standardized ways of measuring the attribute of software processes, products and services. For example, the Goals-Questions-Metrics (GQM) approach [1] has been widely used toselect metrics based on their relation to overall organizational goals. Instead of collecting metrics at random, measurement is based on the goals of a project. To determine if goals are being achieved, a set of questions is constructed related to each goal. To answer each question, metrics are needed. The metrics are organized into this hierarchy to provide continuous assessment of a project's status. The technique is a powerful way to track progress towards project objectives. For over a decade, the Software Engineering Laboratory (SEL) at NASA's Goddard Space Flight Center has significant experience in measuring software processes for over a decade. Their studies have presented the result of collecting valid software engineering data [3]. Data collected at the SEL was based on the changes made to the software during development. It was later followed by evaluating software development processes through analysis of change data. Metrics can be used to understand a software process, evaluate a software product and goals,control resources and products, and predict attributes of a software process in the future. To improve the software process, WISE collects useful data in terms of changes and responses made to issues generated during the normal workflow of a software project. An issue created in the workflow may be an idea, a question, an error, or any other identifiable "chunk" of communication between project participants. An issue is encapsulated in a form and stored in a backend database. Participants can view issues dynamically and generate metrics based on issue properties. Database queries can extract important information relevant to project metrics. Some software metrics represent measures of the properties of a software process. Using the GQM approach, for instance, a metrics program can address many corporate needs including measurement of process maturity, a tools-evaluation database, and trend analysis [11].It is highly recommended to embed metrics tools in the existing development environment. Developers should understand the need for metrics, otherwise they will neither provide accurate data or use the results of metric analysis. Secondly, metrics should be kept close to the developers. This way the developers would be able to access measurements, evaluate them, and take action as part of standard operating procedure and without hindering schedule or budget. WISE metrics are generated implicitly and can be viewed at any stage of the project by all developers. A separate metrics group is not present that collects metrics and analyzes them. The time devoted to metrics collection and analysis is minimized by WISE. The front end of WISE is kept very simple so that developers need not become experts in measurement theory. WISE generates analysis based on many aspects of a workflow model that are relative to the changes in issue forms. This avoids burdening users with the task of metric collection. It is important to remember that metrics can only show problems and give ideas as to what can be done. It is the actions taken as a result of analyzing the data that yield improvements to the process. This is the reason why it is critical for metrics users to understand that measurement is not the goal. The goal is improvement, through measurement, analysis, and feedback [6]. In [6], a practical view of software measurement that formed the basis for a company-wide software metrics initiative within Motorola has been described. The software process program must be defined in a precise powerful and rigorous formalism. Such an environment becomes a vehicle for the organization of tools for facilitating development and maintenance of the specified process [10].

WISE is programmable and can be customized to the specific process of an organization, but such processes are specified in terms of behavioral descriptions of people, roles, and form-based information. Such an approach yields an incremental, bottom-up definition of an organization's processes. There are many advantages to using behavioral descriptions to specify software process models [13]. The behavioral approach describes software development as a collection of activities or processes which may take place concurrently. This approach leads to better automation, message passing for communication, and provides greater visibility for software processes within an organization. For instance, one should try to describe the software process in terms of the events that occur during the development effort rather than changes to the product. Metrics should be based on analysis of these events. WISE addresses this by logging events in order to track issues throughout their lifecycle. Other software process tools, like Marvel [8], define processes from the top-down: they help in defining the detailed software configuration process by informing users which components are potentially affected before performing subsequent editing, compiling, and other coding activities. Tools such as Marvel assist in development and maintenance efforts through controlled automation, but do not address the source or motivation of changes throughout the lifecycle. In our deployment of WISE, we have been careful to integrate it carefully into existing process environments. Many software tools play an important role in the software development process but use of a tool within a process changes the process itself. For example, some tools are used by users taking on a specific role. Other tools are used by users in multiple roles, during many activities and for processing documents of multiple users. Inserting an tool could have a significant impact on the development process. In order to control the insertion of the tool a method called "Tool Insertion Method" [5] has been proposed. The key elements of TIM are tracking the progress of a tool used to improve the process. Indeed, we have used WISE in the development of WISE itself to study the effects of such changes. The issue-based approach used in WISE is dominant not only in problem tracking, but is critical in capturing design rationale, informal information, requirements, and change requests.

Design dependencies can be represented in a issue-based style [9] and tracked throughout the lifecycle of a project. In all cases, participants in an issue-based discussion contribute their expertise and viewpoints to discover and resolve issues. Each issue is followed by one or more positions that respond to an issue. The issue-based model is now almost 20 years old. There is a clear need for automated tools for software management and measurement created by a focus on understanding, managing, and improving the software process.

1.4 Conclusions

WISE is an automated system that helps in software project management and measurement. We have described how such tools can help in improving software development processes through tracking and measurement of change activities on project issues. From our practical experi- ence with WISE on several projects, we believe that such automated tools hold much promise for providing realistic assessments of software development projects particularly in large-scale projects with verification and validation contracts. While our use of WISE has been successful, many issues remain problematic including:

1.4.1 Metric frameworks

More work is needed to construct metric hierarchies for organizations based on specific strengths, weaknesses, and corporate policies. For example, the skill levels of programmers in different groups within the same company vary widely. Such differences have a profound effect on the ability of measurement tools to predict performance.

1.4.2 Security and privacy

These are a major concern in WISE. Several studies [4] have found high participation in automated measurement project by users who feel that they control access to workflow information. Users that feel they are being watched and judged by management will not use or they will circumvent such tools. From our experience, we also feel that measurement should focus on the individual user and let the user control the permissions and visibility of their workflow transactions. WISE allows users to define history visibility so that even managers cannot access change data without explicit permission from the user who owns the data.

1.4.3 Process validation

The bottom-up approach to process definition through forms and roles sometimes creates "black holes" in the process where issues can remain unresolved. We have considered the use of finite-state machine model checking tools to find incomplete and inconsistent paths in the composite process as a part of the WISE system.

Although the definition of an organizational process through roles and the workflow of electronic forms may be indirect and imprecise, WISE provides a flexible platform for accommodating the processes changes. For instance, on several projects it is only after an initial prototype that we discovered that new roles were needed, additional issue fields had to be added, and new field had to be added to forms in a project. This incremental approach, however, is much more productive that trying to define a complete and consistent process from the top-down. Indeed, such changes reveal the progress (or lack of) in efforts to improve development processes. We continue to examine this "meta-view" of process improvement as well as collecting and analyzing results of current projects using WISE.

References

[1] V. Basili. Software modeling and measurement: The goal/question/metric paradigm. Technical Report CS-TR-2956, University of Maryland Computer Science Department, College Park, Maryland, September 1992.

[2] V. Basili. The experience factory and its relationship to other improvement paradigms. In Proceedings of the 4th European Software Engineering Conference, Garmish-Partenkirchen, Germany, September 1993.

[3] V. Basili and D. Weiss. A methodology for collecting valid software engineering data. IEEE/ACM Transactions on Software Engineering, SE-10(6):728-738, November 1984.

[4] M. et al. Bradac. Prototyping a process monitoring experiment. In Proceedings of the 15th International Conference on Software Engineering, pages 155-165, May 1993.

[5] T. Bruckhaus. Tim: A tool insertion method. In Proceedings of the 1994 CAS Conference, October 1994.

[6] M. Daskalantonakis. A practical view of software measurement and implementation expereinces with in motorola. IEEE Transactions on Software Engineering, 18(11):998-1010, November 1992.

[7] W. Humphries. Managing the Software Process. Addison-Wesley, sei series in software engineering edition, 1989.

[8] G. Kaiser and P. Feiler. Intelligent assistance for software development and maintenance. IEEE Software, pages 40-49, May 1988.

[9] M. Lubras. Representing design dependancies in an issue-based style. IEEE Software, pages 81-89, July 1991.

[10] L. Osterweil. Software processes are software too. Communications of the ACM, May 1987.

[11] S.L. Pfleeger. Lessons learned in building a corporate metrics program. IEEE Software, pages 67-74, May 1993.

[12] L. Westfall. Software metrics that meet your information needs. In Proceedings of the 4th International Conference on Software Quality, October 1994.

[13] L. Williams. Software process modeling. In Proceedings of the 10th International Conference on Software Engineering, pages 174-186, April 1988.

Phase 3Report

Task 2.2

1. Task 2.2 -Collaborative Environment for IV & V

The efforts of this group focussed on the collaborative environment for IV & V (see [1]) and specialized tools and approaches for program development and IV & V.

1.1 ISS

Any Collaborative Environment Software Package should be based on some underlying database. It is not only because of the large amounts of data which need to be retrieved, but also for managing this data. We also should take into consideration the heterogenous character of databases involved in the work of some distinct teams —as in the case of IV&V. All those aspects bring the necessity of some distributed database manager to support access to heterogenous databases in a uniform manner. This service is provided by the Information Sharing System (ISS) developed earlier at CERC.

ISS supports different database management systems such as Oracle, Versant, and FoxPro. It runs on different platforms: SUN, DEC, SGIs. This is convenient from the user point of view, because each member in the IV&V team can use their machine to browse through the software product. Also, each team member can store their data (code files, test results, observations, reports) in their database, and all the others team members can access the information in the same way, as if their own. There are a few drawbacks of the ISS which are currently being addressed. These fixes will be available in the future releases of the ISS. Some of the drawbacks are as follows:

- ISS does not provide a way to update information in the individual databases. This means that the ISS is only a browser of the information, and the updates should be done off-line.
- ISS does not provide a dynamical model. That is, ISS has a statically defined model which is loaded into the server. This can be a problem in an evolving IV & V scenario because the ISS will be not be able to track the modifications to the model.

Using the ISS we have built a software product model (see Figure 1). The ISS can keep track of all releases of the software product. The evolving product model is stored in the Project Coorination Board (PCB). This is different from the static model in the ISS. Currently, we are studying the issue of compatbility between the two models. The ISS sees the model of the product as a generic one, giving the possibility to browse through different versions of the code, retrieve files and display them, the PCB needs a model to describe the functionality of a software product. That is, in PCB we need to know what functions are implemented (and by whom), we need a way to specify constraints on different parts of the code. The two approaches can be unified. In order to achieve this we decided to use the model in the Teamwork CASE tool that we have and build a gateway into Teamwork. The Teamwork tools provide a variety of structured and display oriented (Postscript) information about the software being developed. Based on this information we can build a bridge between the generic model of ISS and the specific one of the PCB.

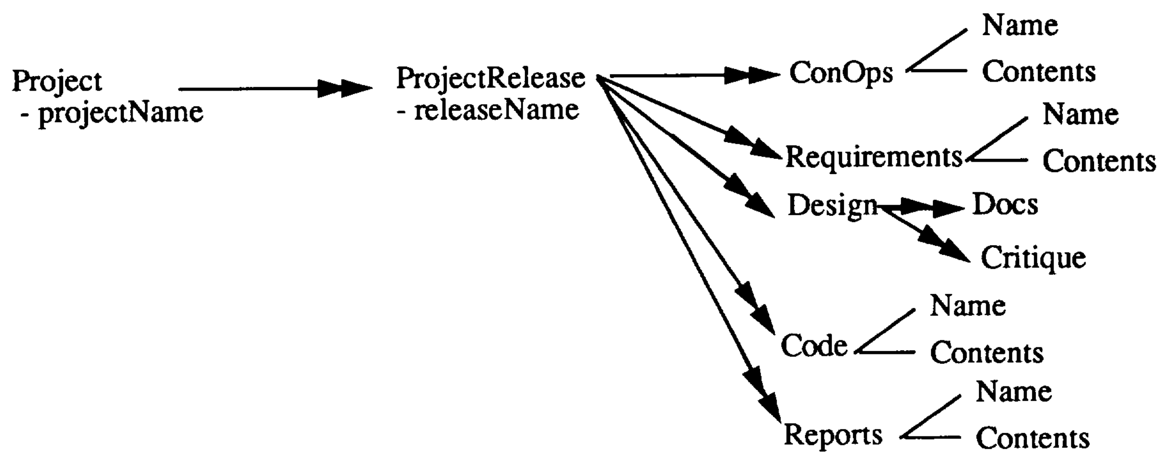


Figure 1: Software product model in the ISS.

Consider the software product model given in Figure 1. We kept this model very simple in order to make it as general as it could be. Now, for PCB, we should take an instance from this model, say one of the ProjectRelease instances, and extract the functional information from it. This can be the model to feed into PCB. This information can be extracted from Teamwork using the Teamwork Gateway.

1.2 New ISS

The ISS developed in the previous phases integrates heterogeneous and distributed databases. However, the mechanisms used to exchange objects are adhoc and not based on a well-established standard. In this phase, we are developing a New ISS based on the CORBA standard for the exchange of objects. This effort is being developed jointly with funding from ARPA in the DICE project as well. The following are the architectural modules of ISS

1.2.1 Interoperability and information sharing

The Information Sharing System (ISS) provides access to information in diverse format and systems. In order to effectively to deal with heterogenous legacy environments, there is critical need to be able to interoperate. The specific requirement is to be able to communicate to these diverse repositories in some standardized ways. In earlier version of this system, we adopted our own implementations of a uniform way to interface to different servers running on different machines. In this current version we have adopted the CORBA specification for server level interoperability. We are also supporting the http-protocol as a mechanism to support client-level interoperability. Architecture for information sharing
Figure2 highlights the various components associated with ISS. version 3.0. The various components associated with this figure are explained below.

1.2.2 Interface Manager

This module is responsible for interfacing with the Mosaic-compliant client on one side and the CORBA-compliant server on the other side. The following are the responsibilities of this module:

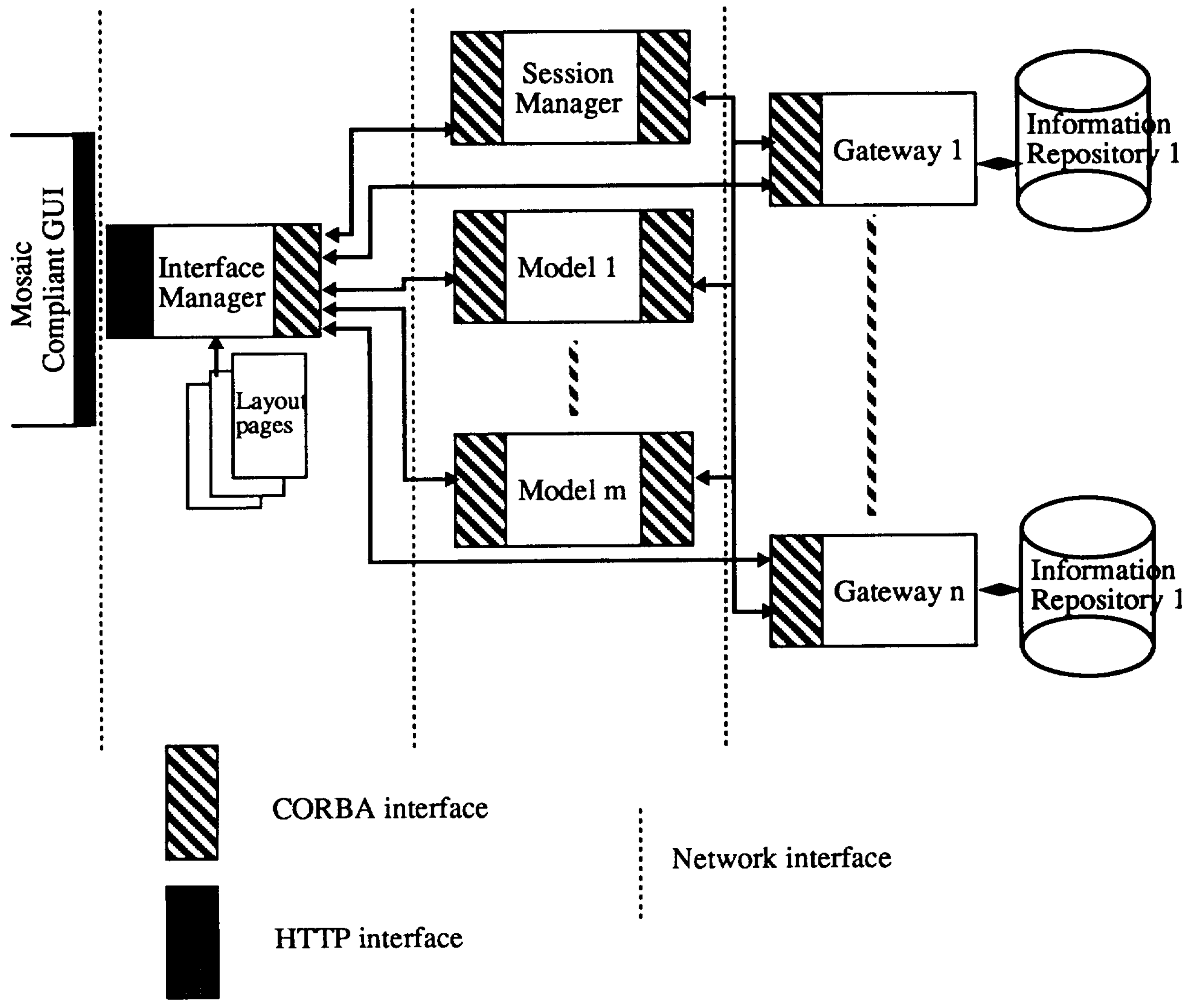


Figure 2: Architecture of the New ISS

to handle logins
to translate Unified Resource Locator (URL) requests from Mosaic clients to document pages.
Logins are handled by this module by validating the user name and password using standard Unix mechanisms.
The URL translation process are handled by a combination of state information sent with the URL (i.e. session information), the type of document requested (i.e. flowsheet, popras form, referral form), the layout page associated with the document type, and queries to information servers.
The Interface Manager is stateless and can handle multiple simultaneous queries from multiple users. Sample layout page information shown below where the information in square parenthesis are commands that are interpreted by this module as calls to the information servers.

```
<A NAME=top>
<PRE>
<H2>[SELECT firstname, middlename, lastname FROM demographics where personid
=%SSNumber%] Chart</H2>
```

1.2.3 Session Manager

The Session Manager instantiates a new session for each user who logs into the system. This process involves instantiating a specific set of gateways (such as Oracle gateway and file archiver), setting up sessions to these as the user who has just logged in and instantiating models (see next section) which interface to these gateways. The session manager is also responsible for closing these connections at the time of logout or close these connections using a time-out mechanism.

1.2.4 Gateways

The gateways are Orbix servers that interface to information repositories. The gateways have standardized interfaces but their implementations vary depending on the type of repository they are connected to.

1.2.5 Models

There are a collection of user defined models that specify what types of information that needs to be served by the system. These models could be specific,such as a flow sheet which is a type of model that is needed in the ARTEMIS project, or they could be a little more generic (an example here would be the gateways to Oracle and File Repositories). This version of ISS will have to provide an implementation of these models which function analogous to the mappings in the earlier versions. Currently mappings are coded procedurally. Currently, we are working towards a declarative specification of the mappings.

1.3 Web*

(pronounced "Web star") -- Web* is a technology developed at the Concurrent Engineering Research Center (CERC) at West Virginia University with funding from the Advanced Research Projects Agency (ARPA) to make information on the Web more easily accessible. The Web* software allows the linking of any information source to a Web client such as Mosiac by allowing a person to specify HyperText Markup Language (HTML) or other ASCII-based templates which are dynamically filled in when requested by the user. The templates embed commands in the Tool Command Language (TCL) and are interpreted and can be used to retrieve and dynamically fill the templates with information. Web* comes with interfaces to call clients that comply with the Common Object Request Broker Architecture (CORBA). The current implementation uses Orbix. One of the key features of Web* is that it provides mechanisms to deal with the stateless nature of the HTTP protocol. Web* is intended to be used by a basic Web client that knows only "html". Thus, the Web server produces html for use by the client. The main programmability in Web* comes from TCL. Currently Web* ties into ORACLE databases using TCL commands.

Web* is designed to help writing dynamic HTML pages. Web* is a CORBA compliant client that uses Tk/Tcl and/or Scheme to make connections to different services offered by Orbix$^{(TM)}$. Orbix is a comercially available implementation of CORBA standard. Since in our previous efforts we developed a series of services for the ISS project, such as Oracle gateways and File Archiver gateways. We now want to make them accessible through World Wide Web.

We extended Tk/Tcl and Scheme to support primitives that can dynamically access Orbix Services. Thus, our project model can be very easily be stored in any distributed database system and users all over the world (if permissions are granted) can access and/or update information for the project. This approach enables a real cooperative environment to be developed and customized (due to its simplicity) by the users themselves. Almost everybody can write a HTML page. Web* pages are not much more difficult to write and they provide the user with the power of writting pages that contain real dynamic data.

Our approach, from the IV&V perspective, is to create a model for a project life cycle and store the information into a variety of databases (each team involved in the IV&V process may have their own DBMS), and provide a template of Web* pages that allow a user to access the information in the distributed databases.

This tool, together with the other cooperative environment tools (such as Monet and Comix) provide a team in the IV&V process with full colaboration capabilities, while they still can use the other tools needed (Software Analysis packages) transparently.

1.4 Information Sharing System Gateway to CADRE Teamwork

Consider the following scenario:

Some members of the IV & V team are working in a CADRE Teamwork environment and doing requirement analysis, design and modeling. The information pertaining to requirements analysis, design and modeling is stored in the Teamwork database on one host, and a team member working in some other area, (e.g. testing) needs to access this information.

Since the team member has access to the ISS, he/she can access the information if there is a gateway from the ISS to Teamwork. Obtaining results from the Teamwork gateway involves the following steps:

. Develop a model of the data in Teamwork
. Obtain user query through ISS
. Parse the query in ISS
. Translate the query method to a query method in Teamwork gateway
. Retrieve the information from Teamwork gateway
. Convert the results into the ISS format

Figure3 illustrates the architecture of the Teamwork gateway for ISS. We have developed all of the above and illustrated the gateway using the Polygons scenario that we are working on. The gateway can present results in Postscript, EPS, Interleaf, Framemaker or ASCII formats. Entities such as Process Specifications, Module Specifications, Transition diagrams, Structure charts, and Data flow diagrams can be retrieved. We use the Teamwork Toolkit to access the information inside the Teamwork database. After the ISS query is parsed, we reconstruct the query in Teamwork Toolkit format and submit it to Teamwork. The result of the query is stored in a Virtual Instance Collection object. This is the object representation in ISS. We browse it using the ISS model browser.

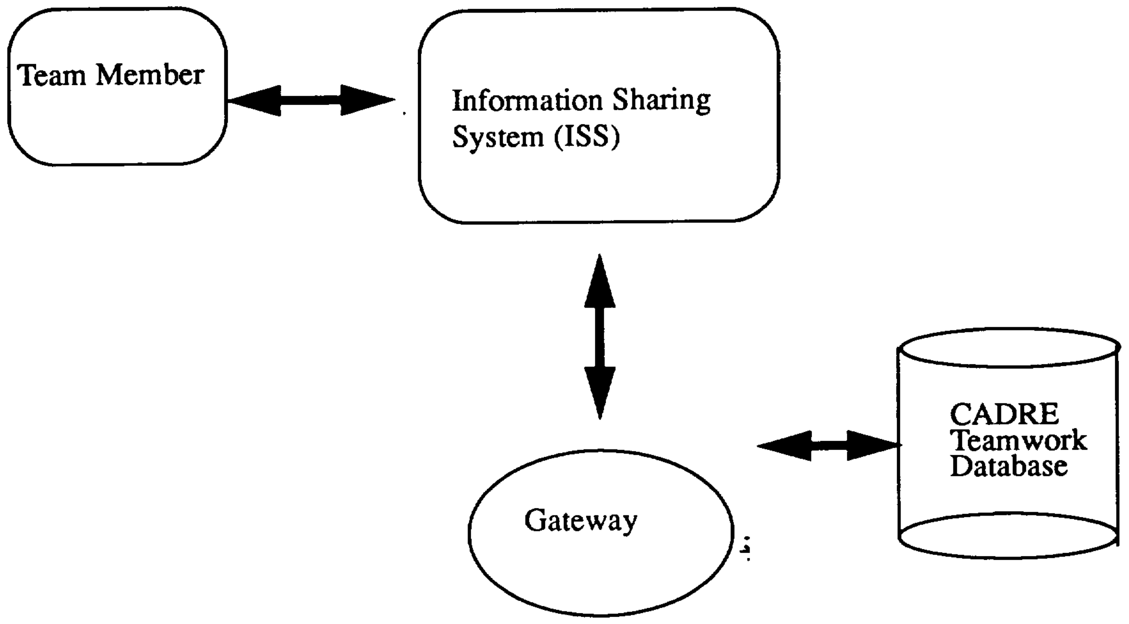


Figure 3: Teamwork Gateway for ISS

1.5 Collocation and Application Sharing in a Collaborative Environment -- NEW MONET

Team members in the IV & V project often need to consult with each other. For example, several members of the IV & V team can get together to evaluate software from various perspectives and write a joint report describing their findings. The ability to engage in desktop conferencing over a computer network using audio, video and graphics is provided by the MONET system. MONET also captures the minutes of the meeting for archival and subsequent use. The COMIX component of MONET enables any X-windows applications to be shared by a group of people. Thus COMIX enables a team of people to share a computer application or jointly author a document.

1.5.1 Problem Statement

While different versions of MONET has been successfully developed on various workstations such as Digital, SUN and SGI machines, two crucial factors impede the ubiquitous use of MONET and MONET-like systems (such as SHOWME from Sunsoft, Invision from Silicon Graphics, Proshare from Intel and others). The first factor is the lack of interoperability between these software modules running on different hardware and Operating system architectures (i.e., A SHOWME version running on BSD UNIX will not work with a version running on SOLARIS and there is no interoperability between different audio/video hardware). The second factor is the non availabilty of sufficient bandwidth on most networks. We will make MONET functional on networks with great promise such as ATMs. The focus of the new MONET implementation will be to provide solutions to alleviate both these two impediments.

1.5.2 Approach

We intend to address the interoperability issue by structuring the existing MONET system to support a Virtual Machine architecture which can be easily mapped to different hardware and operating system architectures. By seperating the architecture into device/operating system dependent and device/OS independent parts in addition to embracing defacto standards such as mu-law for audio, mpeg for video, and using an interoperable standard such as CORBA one can design an interoperable desktop conferencing system.

To alleviate the bandwidth problem, we intend to use new technologies such as ATM over telephone lines (available in Jan 1995, from First Virtual of Santa Clara) as one of the lowest supported physical infrastructures for the future versions of MONET.

1.5.3 Statement of Work

The MONET system has been redesigned to support the MONETVM architecture. Already a version which works on different video/audio hardware (Parallax, Sunvideo etc.) is in the initial prototype stage. This prototype will be thoroughly tested to ensure the stability of operation. We are now implementing the MONETVM on PCs executing the Windows 95 Operating System and native video hardware such as Intel's Smart Video Pro (based on Indeo technology). The video software is Microsoft Video for Windows compatible and will work with any video board that is Video for Windows compatible.

Subsequently we want to ensure the interoperability between Windows 95 based MONET and the UNIX based MONET working with different audio and video components. We have already achieved it for all components except video.

Finally we intend to develop IDL specifications for each of the Multimedia objects in IDL and design the object lifecycles based on CORBA infrastructure.
Finally at the infrastructure level, we intend to experiment with infrastructures such as ATM over telephone lines to ensure ubiquity of the system.

1.5.4 Progress

1.5.5 MONET Conference Manager

We redesigned and implemented an interoperable conference manager in C++ using NIH classes. The conference manager uses an object oriented design and defines clean messaging interfaces with the audio, video and application sharing servers.

The PC version uses ACE(Adaptive Communication Environment) for encapsulating network related functions. Inter module messages are sent and received using the Windows 95 API.

A log of events regarding connections made, attempted and refused, along with other major events is maintained.

1.5.6 Graphical User Interface on the PC

Currently, we are also developing user interfaces for the PC platform.

We used Microsoft Visual Basic 3.0 to build a prototype of MONET conference system on PC running Windows 3.11. Our plan is to build the system on a PC running Windows 95, since the operating system of Windows 95 supports multiple processes, multiple threads and 32 bit addresses, and we believe it will be the new standard in the PC world. We are currently transitioning to Microsoft Visual C++ to write the graphical user interface which will then be linked to the conference manager and the audio and video virtual devices.

We have built a prototype of the Windows 95 version of MONET using Microsoft Visual C++ 2.0. The multi-threaded nature of the Windows 95 system allowed us to provide separate threads for audio, video and whiteboard modules.The GUI resembles in spirit to most of Windows applications and has been desgined to facilitate smooth operation and control of the entire MONET. The audio and video have been seamlessly integrated into the interface with options for configuring parameters and monitoring network statistics.

1.5.7 Device Independent Audio and Video Managers

Now, device independent video an audio managers have been implemented and the video manager currently works with Parallax and SunVideo boards on Sun SPARC platform and any Video for Windows compatible video board on a PC runing Windows 95. A new graphical user interface has also been implemented on Unix systems for MONET which supports intuitive actions.

The audio now supports full duplex functionality and also has the ability to play and record audio files. It is also interoperable between the UNIX and PC platforms. The video is capable of using the Connectix as well as the Sun cameras.

1.5.8 MONET Transcript

The idea of the transcript is to record a Monet conference so that users can have access to older conferences whenever they want to refer to what happened in previous conferences. This is especially useful in multi-user conferences where users can have access to the proceedings of the conference on-line. The conferences will be archived and can be retrieved using any Web-Browser available.

As Monet is a multimedia conferencing system, the transcript includes all forms of multimedia documents such as text, images, audio and video. Each conference transcript resides with each user and he can add anything he wants in his transcript. Each transcript is therefore a personal document.

The text part of the transcript can be the *text talk* going on in the conference, an agenda file if necessary and any text file the user wishes to include.

The image part of the transcript is the snapshot of any part of the screen she is using. A grid appears on the screen which can be selected from any point on the screen and dragged to any point and the resulting window is stored as an image in the X window dump format. The user can use the X window undump format to see it again.

The audio part is recording the audio as a Sun *.au* file. This is handled by the *nevot* package which is currently being used in Monet audio. We just have to send commands to the nevot server to record audio and stop it given a filename. The nevot server records both incoming audio and the local user audio coming from the microphone.

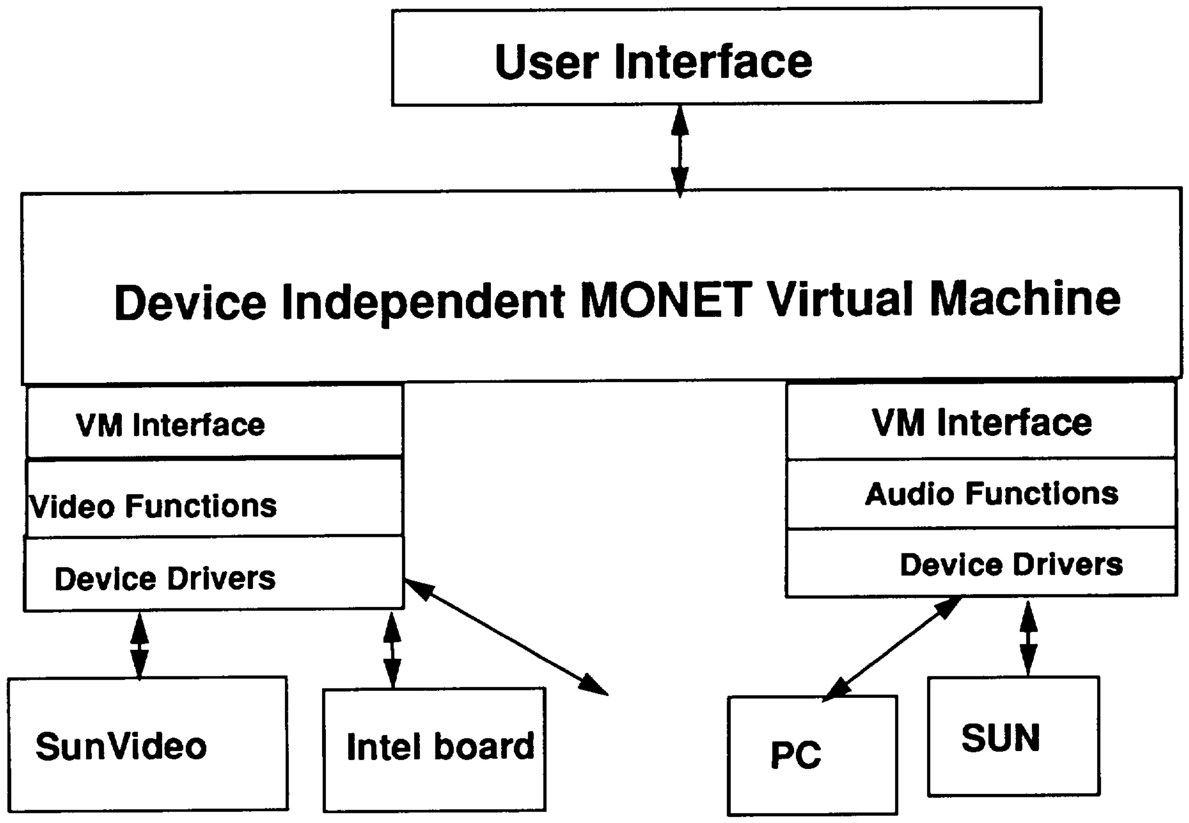
The video part is yet to be implemented. It involves storing a video clip as an MPEG file when the user selects one of the incoming video streams from the network or her own camera.

The problems encountered are the huge sizes of the audio and video files. We are working on a solution to limit the sizes so that the conference can go on smoothly without crashes.

The next part to be implemented is the file archiving and retrieving part of the transcript. The files are archived based on some events such as date of conference, time of conference, participants, name of the conference, and some important keywords. The file retrieval involves searching the file archiver when the user provides one or more of the above mentioned events and retrieving the appropriate files associated with the search results. This is being done using the new yet-to-be-released HOTJAVA browser since it can be programmed to include display interfaces and can be accessed across the World Wide Web.

1.5.9 PCS Compliance

We are currently studying the Personal Conferencing Specification 1.0 in order to assess how to make MONET compliant with this new and upcoming industry standard. This will impact the design and implementation of MONET on the PC platform. We have also studied the ITU-T series recommendations (T.124,T.125) for audiographic and audiovisual conferencing systems. This will be required for interoperability between MONET and other teleconferencing systems.



1.6 The Polygon Operations Scenario

The IV & V environment we are developing is being tested on a scenario. This scenario is based on a program that performs boolean set operations on polygons. The polygons we consider can contain inner loops or holes (as in swiss cheese) within them. The program we are using performs regularized set operations union, intersection and difference on these polygons. The result of the set operations is displayed on a color monitor using a graphics package called SRGP developed at the Brown university. This program has several features that prompted us to use it as a first choice for the IV & V scenario. The program is small enough to be manageable by our mock IV & V team, but not so small as to be considered trivial. (It consists of over two thousand lines of C code and over 125 functions). The program uses a graphics library for display that was developed elsewhere. It runs on Unix platforms, but it should easily portable to PCs and Macintoshes. Some of the other specifications of this program are:

1. Other than for display functions, the rest of the code must be readily portable.

2. It must consider all possible orientations of the polygons, which usually result in a number of special cases, and must also be numerically robust.

3. It must handle arbitrarily large polygons quite efficiently. Even though we have stated the requirements rather loosely, one can see that this example poses a number of issues such as performance, portability, functional correctness and numerical precision.

The model created for the polygons project includes various aspects such as requirements, design, code, functional test results, performance test results, requirements analysis results, design analysis results and reports summarizing the results of the IV & V team. For each of this, we have two versions of the document. The model we have developed for IV & V based on this example is shown in Figure 1.

We developed a more reliable version of the Polygons Scenario (the polygons scenario was developed in the initial phase of the project) by using the File Archiver developed within the ARTEMIS project. The new version of Polygons Scenario does not rely anymore on the pre-established location of the files, but uses the File Archiver server in order to access all the required resources. This scenario illustrates all the technologies outlined earlier: CORBA, Orbix, Tk/Tcl, WWW, HTML, HTTP, Web*, ORACLE gateway and the File Archiver.

1.7 The Flight Operations Scenario

We developed a new IV & V scenario based on the Earth Observation System Flight Operations in collaboration with other members of the project. This scenario demonstrates the handling of multi-layered projects based on their representation using diagrams. The names of the diagrams are stored into a ORACLE database. Based on this information, we will be able to retrieve the appropriate files using the File Archiver. As one can see, this demo uses the same technologies as the Polygon Scenario Demo: CORBA, Orbix, Tk/Tcl, WWW, HTML, HTTP, Web*, ORACLE gateway and the File Archiver.

References

[1] R. Karinthi, K. Srinivas, S. Reddy, R. Reddy, C. Cascaval, W. Jackson, S. Venkatraman and H. Zheng, *A Collaborative Environment for Independent Verification and Validation of Software*, Proceedings of the the third workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, WV, April 17-19, 1994.